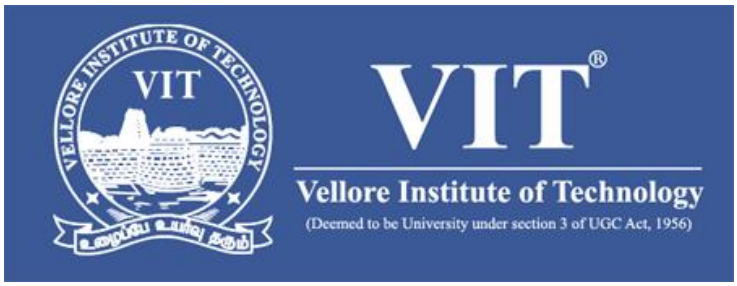
Analysis of You Tube Data Set

Report submitted for the Final Project Review of

Course Name: - Data Mining

Course Code: - CSE3019

Team 4



Prof. Archana T

Team Members

Nimit Agrawal	**16BCE0170**
Poorvit Jain	**16BCE0378**

ACKNOWLEDGEMENTS

This work would not been possible, but it is the wealth of experience and acknowledges that is generated within the portals of VIT University, Vellore which enlightened my path to complete my project.

In performing our project, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this project gives us much Pleasure. We would like to show our gratitude to **Prof. Archana T, VIT University** for giving us a good guideline for project throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in completing this assignment.

In addition, a thank you to seniors and teachers who helped us in the methodology of work, and whose passion for the "underlying structures" had lasting effect.

Many people, especially our classmates and team members itself, have made valuable comment suggestions on this proposal which gave us an inspiration to improve our project. We thank all the people for their help directly and indirectly to complete our project.

Table of content

Introduction

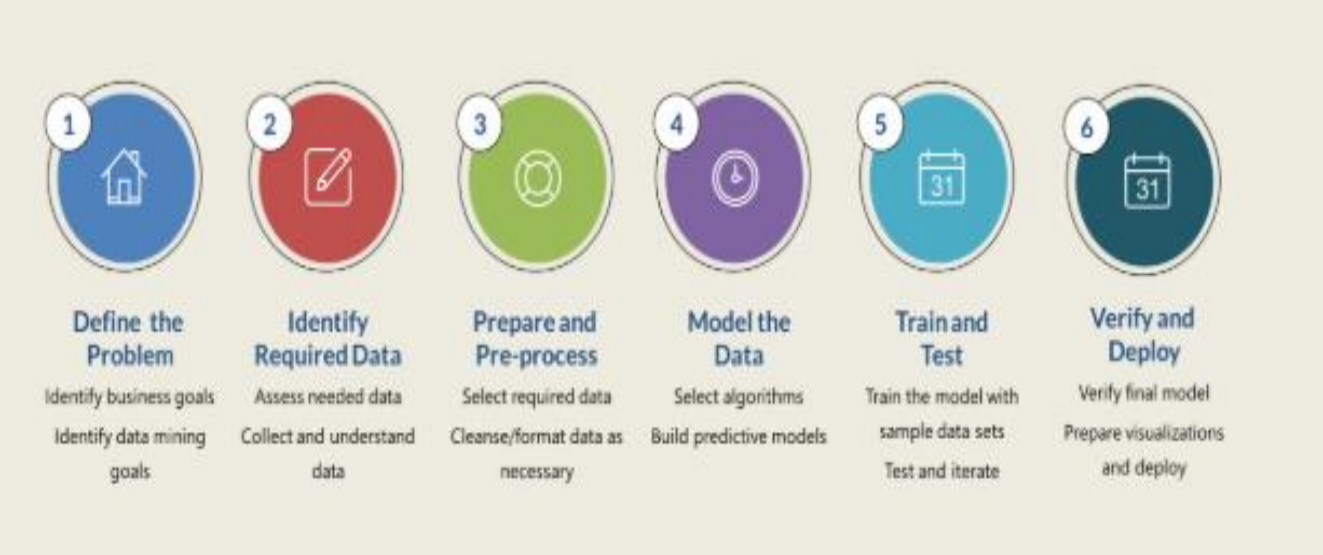
Data mining is the process of finding designs in vast data sets including strategies at the crossing point of machine learning, measurements, and database frameworks. Data mining is an interdisciplinary sub-field of software engineering and measurements with a general objective to extricate data (with astute techniques) from a data set and change the data into an intelligible structure for further use. Data mining is the analysis venture of the "knowledge revelation in databases" process, or KDD. Beside the crude analysis step, it additionally includes database, data pre-processing, model and surmising contemplations, intriguing quality measurements, multifaceted nature contemplations, post-processing of found structures, representation, and web based refreshing. The distinction between data analysis and data mining is that data analysis is utilized to test models and theories on the dataset, e.g., dissecting the adequacy of a promoting effort, paying little respect to the measure of data; conversely, data mining utilizes machine-learning and factual models to reveal stealthy or shrouded designs in an expansive volume of data.

The expression "data mining" is in actuality a misnomer, in light of the fact that the objective is the extraction of examples and knowledge from a lot of data, not simply the extraction (mining) of data. It likewise is a popular expression and is much of the time connected to any type of substantial scale data or data processing (gathering, extraction, warehousing, analysis, and measurements) just as any utilization of PC choice emotionally supportive network, including man-made brainpower (e.g., machine learning) and business knowledge. The book Data mining: Practical machine learning apparatuses and methods with Java (which covers generally machine learning material) was initially to be named simply Practical machine

learning, and the term data mining was included for advertising reasons. Frequently the more broad terms (vast scale) data analysis and analytics – or, when alluding to genuine strategies, man-made consciousness and machine learning – are progressively fitting.

The genuine data mining task is the self-loader or programmed analysis of substantial amounts of data to separate already obscure, fascinating examples, for example, gatherings of data records (group analysis), abnormal records (irregularity recognition), and conditions (affiliation rule mining, consecutive example mining). This generally includes utilizing database methods, for example, spatial lists. These examples would then be able to be viewed as a sort of synopsis of the info data, and might be utilized in further analysis or, for instance, in machine learning and predictive analytics. For instance, the data mining step may recognize different gatherings in the data, which would then be able to be utilized to acquire increasingly exact expectation results by a choice emotionally supportive network. Neither the data accumulation, data planning, nor result elucidation and detailing is a piece of the data mining step, however do have a place with the general KDD process as extra advances.

The related terms data digging, data angling, and data snooping allude to the utilization of data mining techniques to test portions of a bigger populace data set that are (or might be) too little for dependable factual inductions to be made about the legitimacy of any examples found. These techniques can, nonetheless, be utilized in making new speculations to test against the bigger data populations.



The knowledge discovery in databases (KDD) process is commonly defined with the stages:

1. Selection
2. Pre-processing
3. Transformation
4. Data mining
5. Interpretation/evaluation

It exists, however, in many variations on this theme, such as the Cross-industry standard process for data mining (CRISP-DM) which defines six phases:

1. Business understanding
2. Data understanding
3. Data preparation
4. Modelling
5. Evaluation
6. Deployment

or a simplified process such as (1) Pre-processing, (2) Data Mining, and (3) Results Validation.

Data mining can unexpectedly be abused, and would then be able to deliver results which give off an impression of being huge; yet which don't really anticipate future conduct and can't be recreated on another example of data and bear little use. Regularly the outcomes from researching an excessive number of theories and not performing appropriate measurable theory testing. A straightforward form of this issue in machine learning is known as overfitting, however a similar issue can emerge at various periods of the process and in this manner a train/test split - when relevant by any stretch of the imagination - may not be adequate to keep this from occurring.

The last advance of knowledge disclosure from data is to confirm that the examples delivered by the data mining calculations happen in the more extensive data set. Not all examples found by the data mining calculations are essentially legitimate. Usually for the data mining calculations to discover designs in the preparation set which are absent in the general data set. This is called overfitting. To conquer this, the assessment utilizes a test set of data on which the data mining calculation was not prepared. The scholarly examples are connected to this test set, and the subsequent yield is contrasted with the ideal yield. For instance, a data mining calculation endeavoring to recognize "spam" from "real" messages would be prepared on a preparation set of test messages. When prepared, the scholarly examples would be connected to the test set of messages on which it had not been prepared. The exactness of the examples would then be able to be estimated from what number of messages they effectively group. Various factual techniques might be utilized to assess the calculation, for example, ROC bends.

On the off chance that the educated examples don't satisfy the ideal guidelines, in this way it is important to rethink and change the pre-processing and data mining steps. In the event that the educated examples do satisfy the ideal guidelines, at that point the last advance is to translate the scholarly examples and transform them into knowledge.

Dataset link:

https://www.kaggle.com/datasnaek/youtube-new/home

Problem Statement:

1. Finding Category of a new video so as to maximize the number of views.

2. Clustering the dataset based on their category and country.

3. Dependency of one category on another.

Algorithm Used:

1. Decision Tree Algorithm

2. K-means Algorithm and Heirarchical algorithm

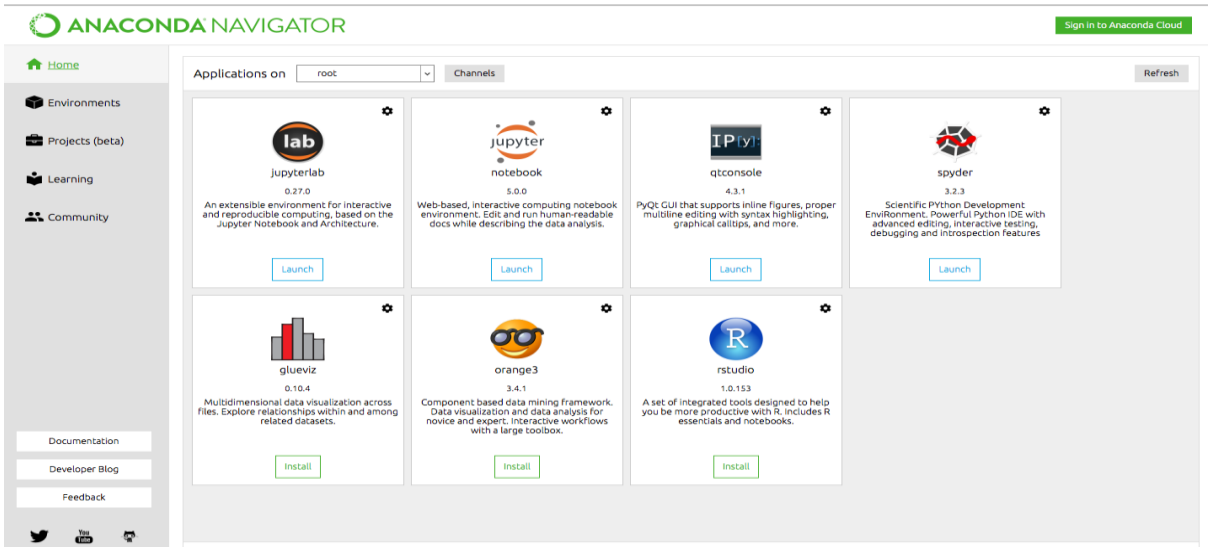
3. Apriori Algorithm

Dataset Description: This dataset includes several months (and counting) of data on daily trending YouTube videos. Data is included for the US, GB, DE, CA, and FR regions (USA, Great Britain, Germany, Canada, and France, respectively), with up to 200 listed trending videos per day. Each region's data is in a separate file. Data includes the video title, channel title, publish time, tags, views, likes and dislikes, description, and comment count.

Tools Used:

1).Spyder

Spyder is an amazing logical condition written in Python, for Python, and planned by and for researchers, designers and data investigators. It offers a one of a kind blend of the propelled altering, analysis, troubleshooting, and profiling usefulness of a far reaching advancement



device with the data investigation, intelligent execution, profound examination, and wonderful representation capacities of a logical bundle.

Past its many implicit highlights, its capacities can be expanded considerably further through its module framework and API. Besides, Spyder can likewise be utilized as a PyQt5 expansion library, enabling designers to expand upon its usefulness and install its parts, for example, the intuitive support, in their very own PyQt programming.

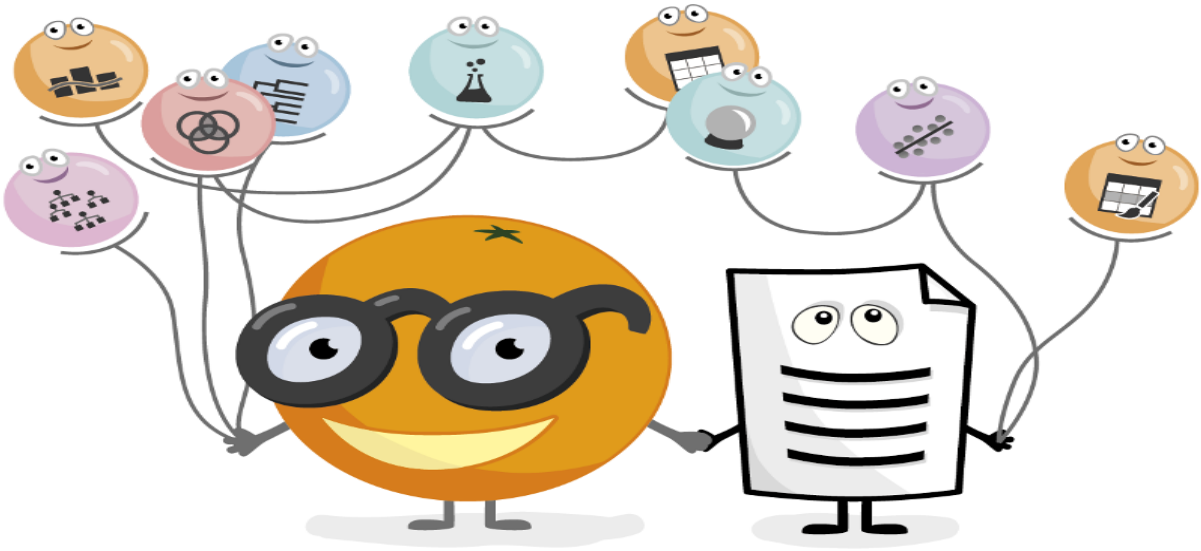
Anaconda Enterprise is an enterprise-ready, secure and scalable data science platform that empowers teams to govern data science assets, collaborate and deploy data science projects.

With Anaconda Enterprise, you can do the following:

- Develop: ML/AI pipelines in a central development environment that scales from laptops to thousands of nodes
- Govern: Complete reproducibility from laptop to cluster with the ability to configure access control
- Automate: Model training and deployment on scalable, container-based infrastructure

2).Orange

Orange is a segment based visual programming bundle for data representation, machine learning, data mining, and data analysis. Orange segments are called gadgets and they go from basic data representation, subset determination, and pre-processing, to exact assessment of learning calculations and predictive demonstrating. Visual writing computer programs is actualized through an interface in which work processes are made by connecting predefined or client structured gadgets, while propelled clients can utilize Orange as a Python library for data control and gadget alteration.

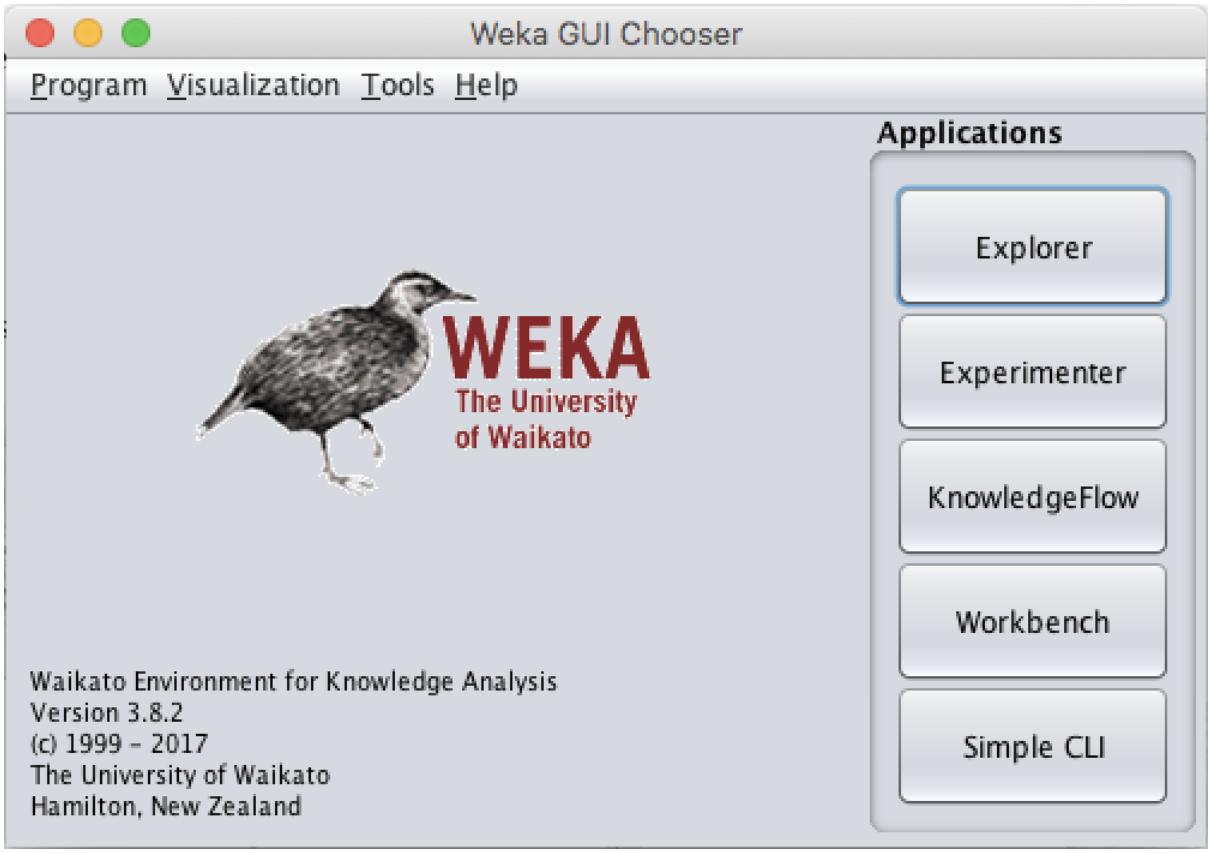


Orange is an open-source programming bundle discharged under GPL. Forms up to 3.0 incorporate center segments in C++ with wrappers in Python are accessible on GitHub. From rendition 3.0 onwards, Orange uses basic Python open-source libraries for logical registering, for example, numpy, scipy and scikit-learn, while its graphical UI works inside the cross-stage Qt system. Orange3 has its own different github.

The default establishment incorporates various machine learning, preprocessing and data representation calculations in 6 gadget sets (data, envision, arrange, relapse, assess and unsupervised). Extra functionalities are accessible as additional items (bioinformatics, data combination and content mining). Orange is upheld on macOS, Windows and Linux and can likewise be introduced from the Python Package Index vault (pip introduce Orange3). As of May 2018 the steady form is 3.13 and keeps running with Python 3, while the inheritance variant 2.7 that keeps running with Python 2.7 is as yet accessible.

3).Weka

Weka contains a gathering of perception apparatuses and calculations for data analysis and predictive demonstrating, together with graphical UIs for simple access to these functions. The first non-Java rendition of Weka was a Tcl/Tk front-end to (for the most part outsider) displaying calculations actualized in other programming dialects, in addition to data preprocessing utilities in C, and a Makefile-based framework for running machine learning tests. This unique adaptation was basically planned as a device for examining data from farming domains, yet the later completely Java-based form (Weka 3), for which advancement began in 1997, is currently utilized in various application zones, specifically for instructive purposes and research. Points of interest of Weka include:



Free accessibility under the GNU General Public License. Versatility, since it is completely actualized in the Java programming language and in this way keeps running on practically any advanced processing stage.

An exhaustive accumulation of data pre-processing and displaying procedures. Usability because of its graphical UIs. Weka underpins a few standard data mining undertakings, all the more explicitly, data pre-processing, grouping, characterization, relapse, representation, and highlight choice.

The majority of Weka's systems are predicated on the supposition that the data is accessible as one level document or connection, where every datum point is depicted by a fixed number of characteristics (typically, numeric or ostensible traits, however some other quality sorts are likewise bolstered).

Weka gives access to SQL databases utilizing Java Database Connectivity and can process the outcome returned by a database inquiry. Weka furnishes access to profound learning with Deeplearning4j. It isn't equipped for multi-social data mining, yet there is independent programming for changing over a gathering of connected database tables into a solitary table that is reasonable for processing utilizing Weka. Another essential region that is as of now not secured by the calculations incorporated into the Weka appropriation is arrangement demonstrating.

Library used with list of functions under each library.

1). numpy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

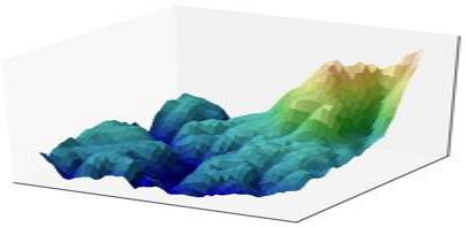
- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

2). pandas

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

3). matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.



Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

4). Copy

Assignment statements in Python do not copy objects, they create bindings between a target and an object. For collections that are mutable or contain mutable items, a copy is sometimes needed so one can change one copy without changing the other. This module provides generic shallow and deep copy operations (explained below).

Interface summary:

copy.copy(x)
Return a shallow copy of x.

copy.deepcopy(x)
Return a deep copy of x.

Algorithm: Decision Tree (P1)

Decision Tree calculation has a place with the group of managed learning calculations. In contrast to other directed learning calculations, decision tree calculation can be utilized for tackling relapse and grouping issues as well.

The general intention of utilizing Decision Tree is to make a preparation display which can use to foresee class or estimation of target factors by learning decision rules construed from earlier data(training data).

The understanding dimension of Decision Trees calculation is so natural contrasted and other grouping calculations. The decision tree calculation endeavors to take care of the issue, by utilizing tree portrayal. Each inside hub of the tree relates to a quality, and each leaf hub compares to a class mark..

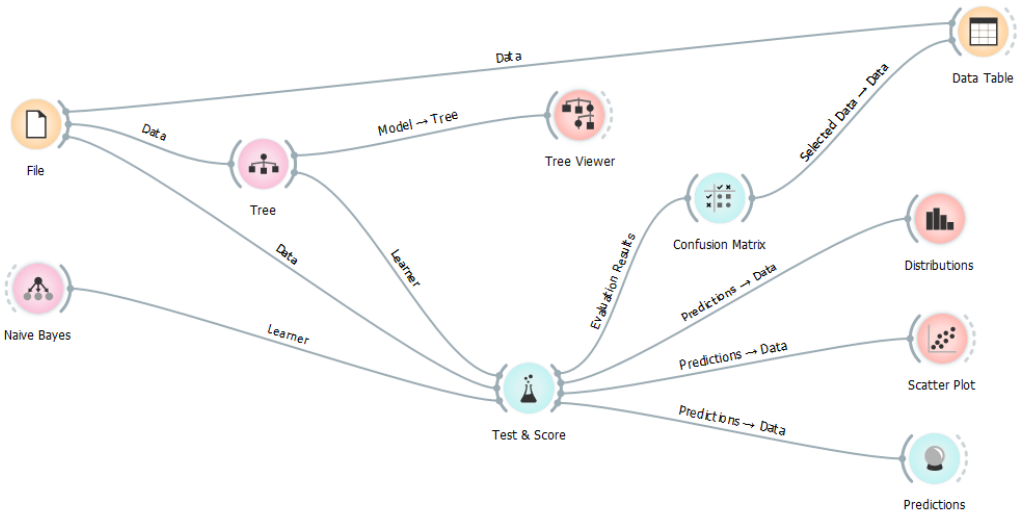
Psuedo-code for Decision Tree

```
INPUT: S, where S = set of classified instances
OUTPUT: Decision Tree
Require: S ≠ ∅, num_attributes > 0
 1: procedure BuildTree
 2:     repeat
 3:         maxGain ← 0
 4:         splitA ← null
 5:         e ← Entropy(Attributes)
 6:         for all Attributes a in S do
 7:             gain ← InformationGain(a, e)
 8:             if gain > maxGain then
 9:                 maxGain ← gain
10:                 splitA ← a
11:             end if
12:         end for
13:         Partition(S, splitA)
14:     until all partitions processed
15: end procedure
```

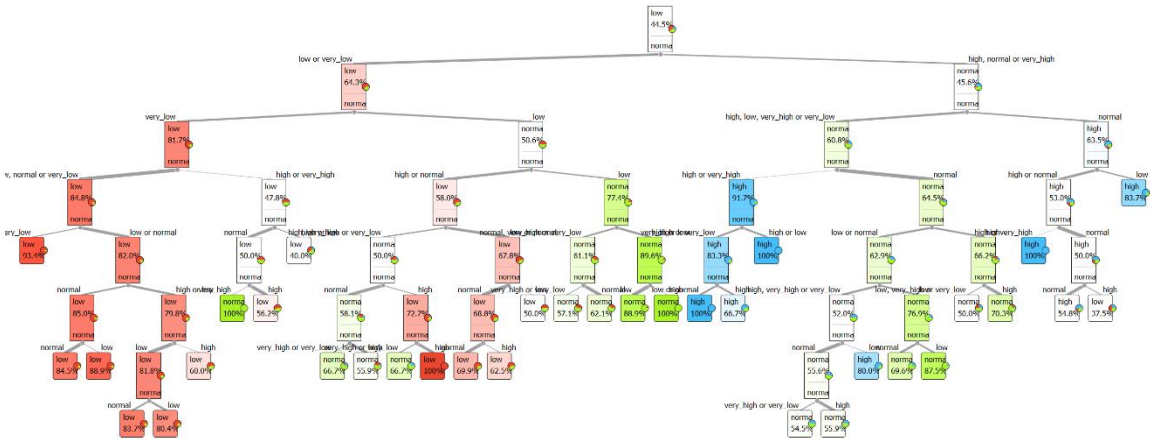
Sample Output Screen for Decision Tree (P1)

Orange tool

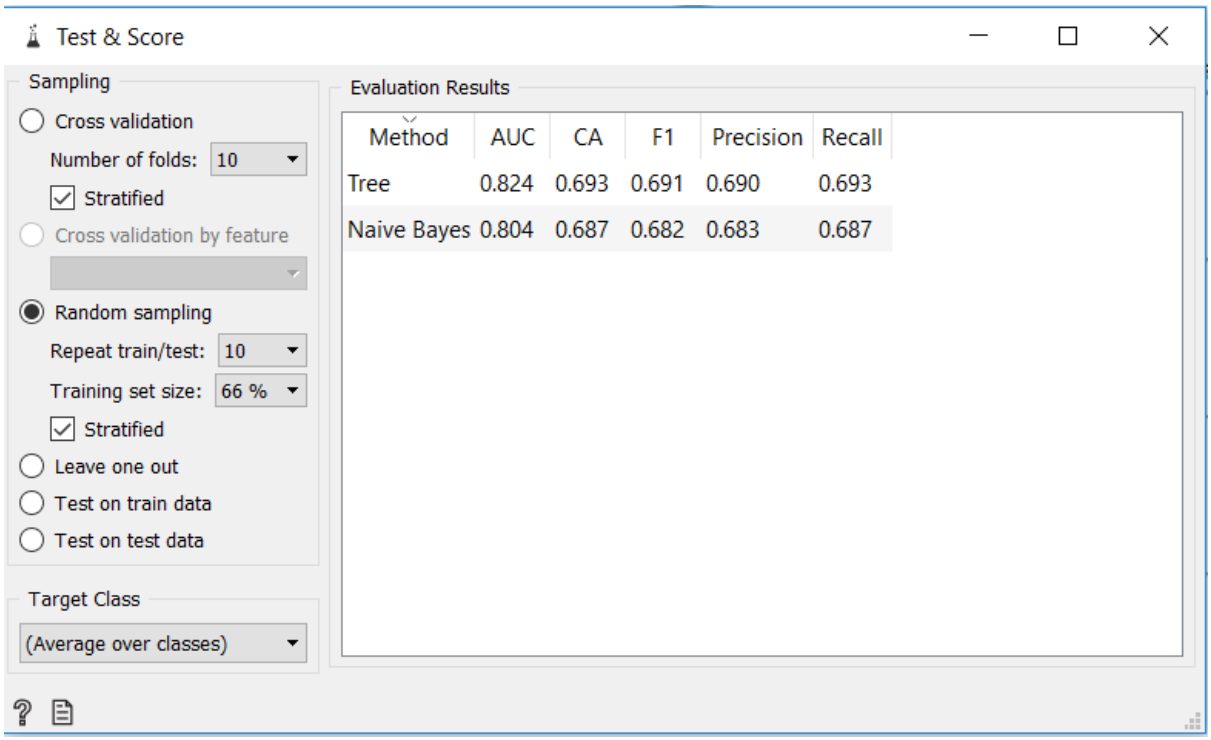
Architecture



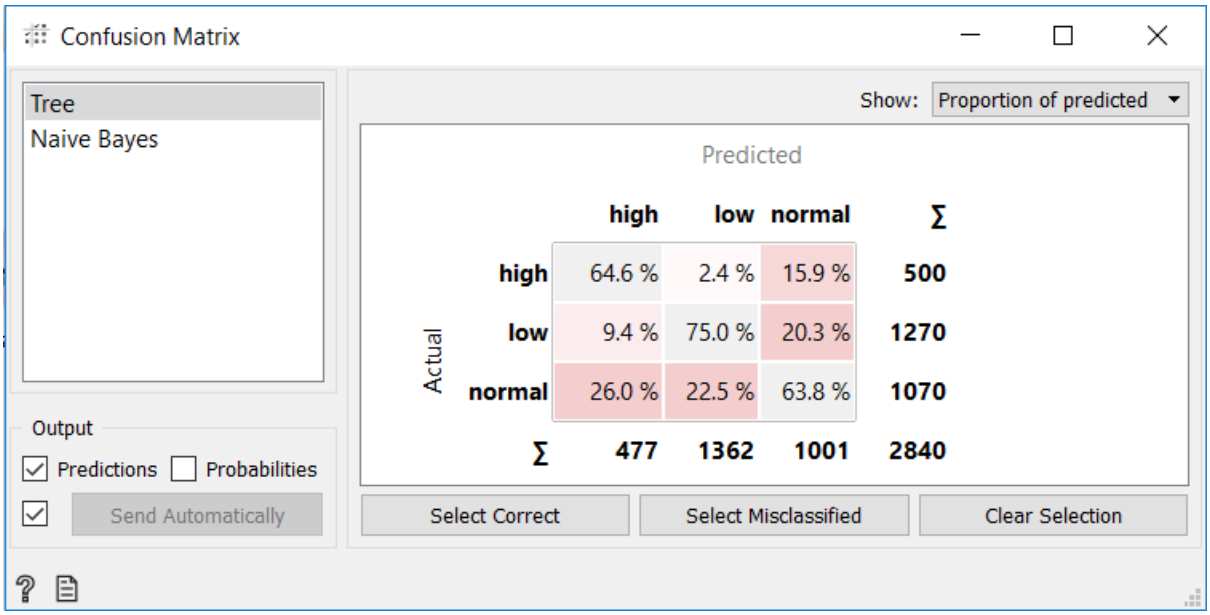
Tree Diagram



Test & Score

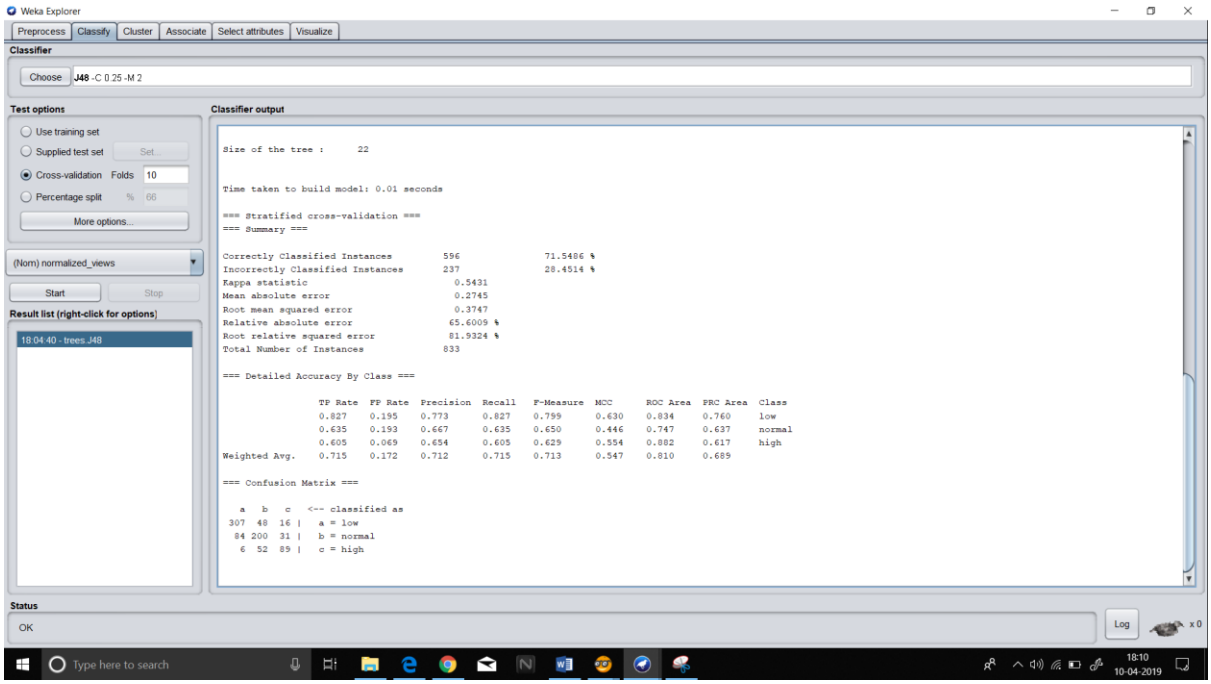


Confusion Matrix

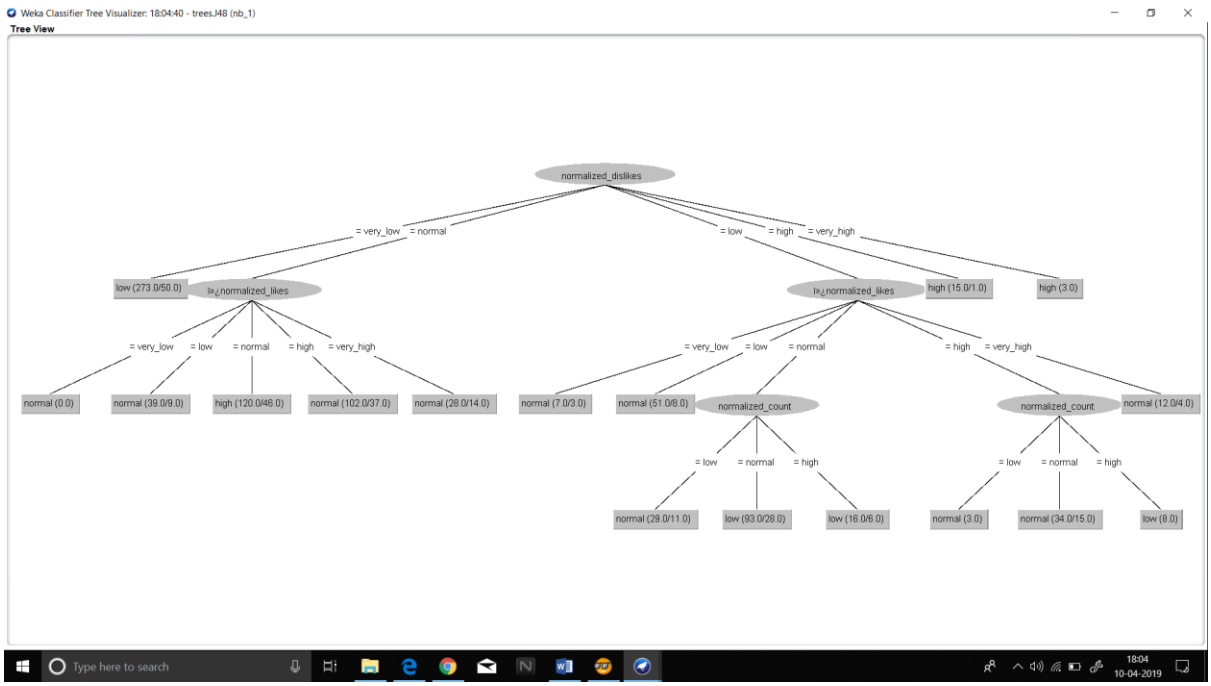


Weka tool

Confusion Matrix



Tree Diagram



Algorithm: K-means algorithm (P2)

k-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modeling. They both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

Algorithm 1: K-Means Algorithm

Input: $E = \{e_1, e_2, \ldots, e_n\}$ (set of entities to be clustered)
k (number of clusters)
$MaxIters$ (limit of iterations)

Output: $C = \{c_1, c_2, \ldots, c_k\}$ (set of cluster centroids)
$L = \{l(e) \mid e = 1, 2, \ldots, n\}$ (set of cluster labels of E)

foreach $c_i \in C$ **do**
  $c_i \leftarrow e_j \in E$ (e.g. random selection)
end
foreach $e_i \in E$ **do**
  $l(e_i) \leftarrow argminDistance(e_i, c_j) j \in \{1 \ldots k\}$
end

$changed \leftarrow false$;
$iter \leftarrow 0$;
repeat
  foreach $c_i \in C$ **do**
    $UpdateCluster(c_i)$;
  end
  foreach $e_i \in E$ **do**
    $minDist \leftarrow argminDistance(e_i, c_j)\ j \in \{1 \ldots k\}$;
    if $minDist \neq l(e_i)$ **then**
      $l(e_i) \leftarrow minDist$;
      $changed \leftarrow true$;
    end
  end
  $iter++$;
until $changed = true\ and\ iter \leq MaxIters$;

Sample Output Screen for K-means Algorithm (P2)

A1 | 1

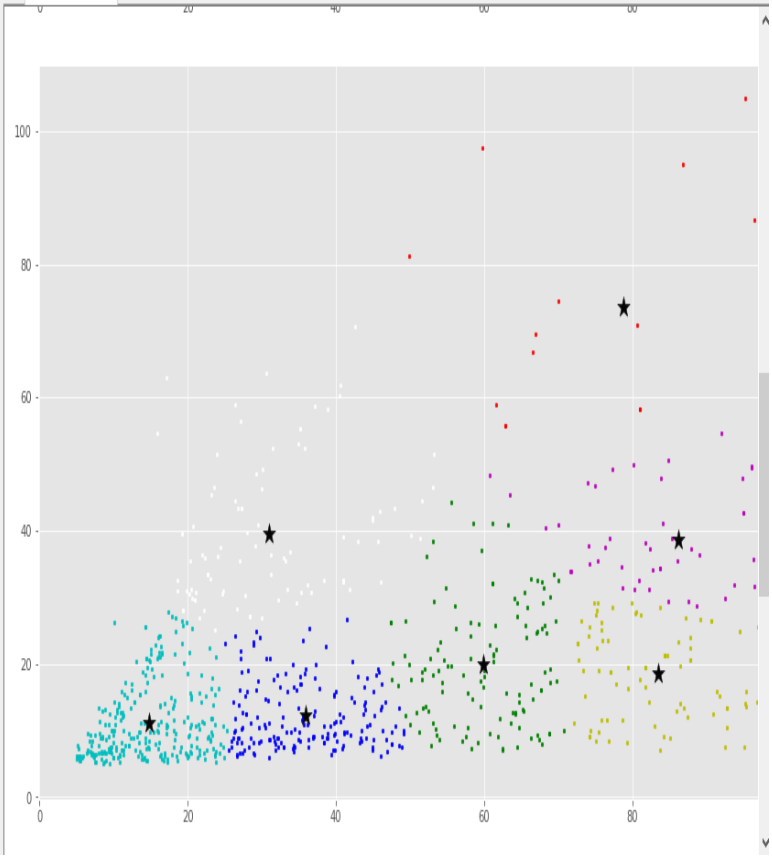
	A	B	C	D	E	F
1	1	1096327	33966	798	882	
2	2	590101	735	904	0	
3	3	473988	2011	243	149	
4	4	1242680	70353	1624	2684	
5	5	464015	492	293	66	
6	6	6106669	98612	4185	4763	
7	7	5718766	127477	7134	8063	
8	8	10588371	132738	8812	10847	
9	9	118223	520	53	23	
10	10	969030	59798	1545	2404	
11	11	632747	4330	2183	2869	
12	12	2348107	32834	710	1743	
13	13	156085	716	53	0	
14	14	472413	2611	250	174	
15	15	836006	24460	180	594	
16	16	89531	238	59	18	
17	17	344545	25717	417	2870	
18	18	35885754	829362	61195	101117	
19	19	209599	14070	448	1105	
20	20	2418783	97187	6146	12703	
21	21	2294242	14960	1361	913	
22	22	1956925	9504	938	956	
23	23	387221	2159	1214	711	

INdia videos

k_means.py | h.py

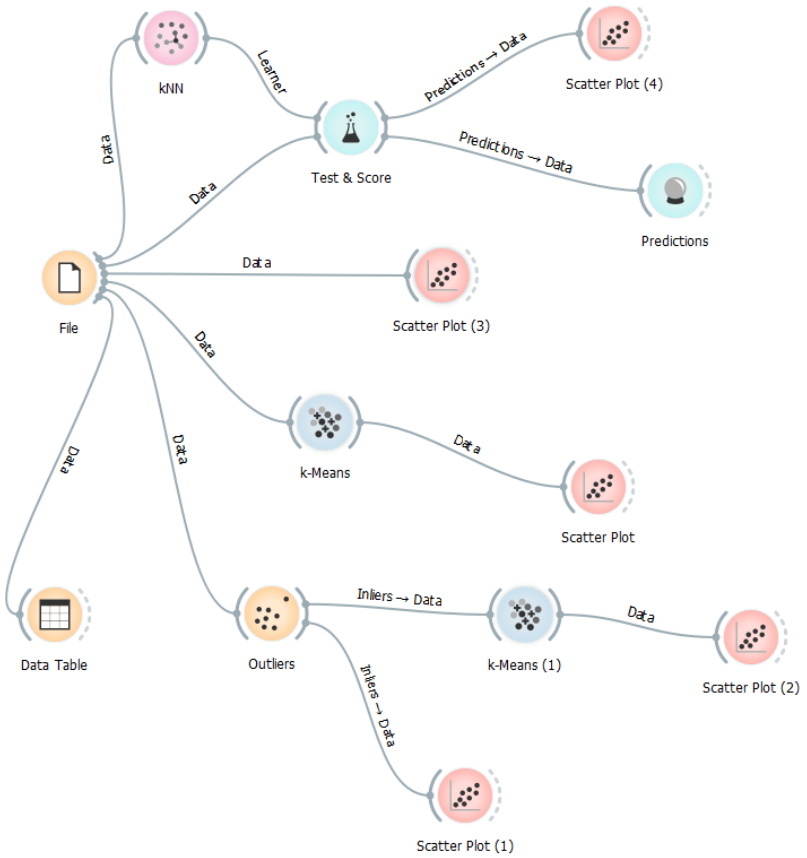
```
C = np.array(list(zip(C_x, C_y)), dtype=np.float32)
print("Initial Centroids")
print(C)

# Plotting along with the Centroids
plt.scatter(f1, f2, c='#050505', s=7)
plt.scatter(C_x, C_y, marker='*', s=200, c='g')

# To store the value of centroids when it updates
C_old = np.zeros(C.shape)
# Cluster Lables(0, 1, 2)
clusters = np.zeros(len(X))
# Error func. - Distance between new centroids and old c
error = dist(C, C_old, None)
# Loop will run till the error becomes zero
while error != 0:
    # Assigning each value to its closest cluster
    for i in range(len(X)):
        distances = dist(X[i], C)
        cluster = np.argmin(distances)
        clusters[i] = cluster
    # Storing the old centroid values
    C_old = deepcopy(C)
    # Finding the new centroids by taking the average va
    for i in range(k):
        points = [X[j] for j in range(len(X)) if cluster
        C[i] = np.mean(points, axis=0)
    error = dist(C, C_old, None)

colors = ['r', 'g', 'b', 'y', 'c', 'm','w']
fig, ax = plt.subplots()
for i in range(k):
        points = np.array([X[j] for j in range(len(X)) i
        ax.scatter(points[:, 0], points[:, 1], s=7, c=co
ax.scatter(C[:, 0], C[:, 1], marker='*', s=200, c='#0505
```

Console 1/A



Orange tool

Architecture



Scatter Plot

Algorithm: Hierarchical clustering (P2)

Hierarchical clustering, otherwise called hierarchical cluster analysis, is a calculation that clusters comparable items into gatherings called groups. The endpoint is a lot of bunches, where each group is unmistakable from one another group, and the items inside each group are comprehensively like one another.

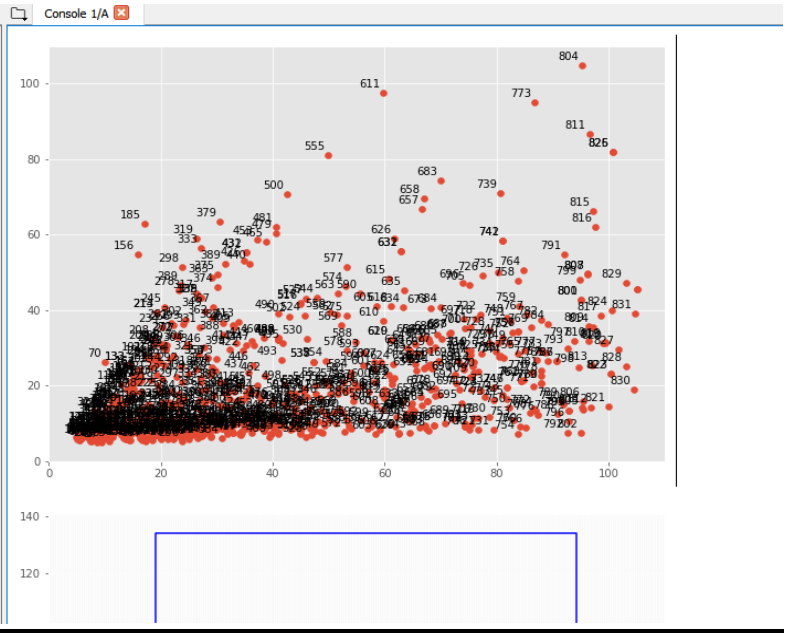
Input: $\mathbf{X}_{n\times d}$ and q;
Output: a collection of q clusterings;

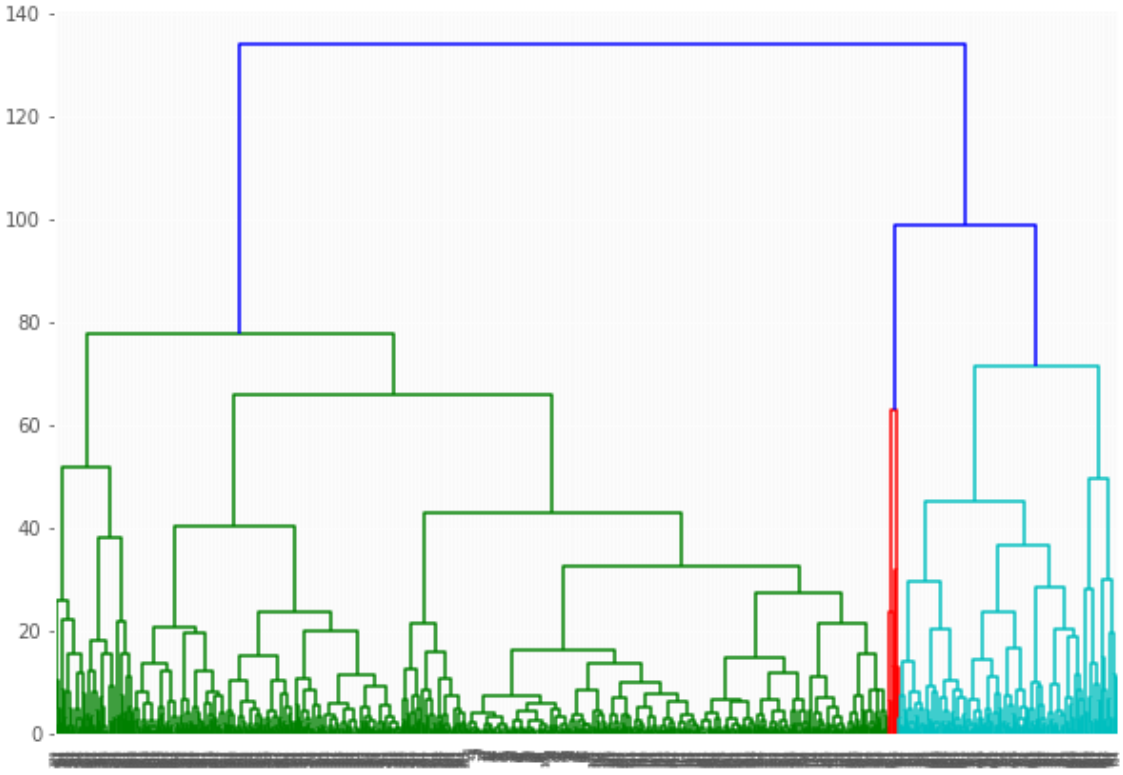
$Y \longleftarrow \text{PCA}(X_{n\times d}, \text{dim}=q)$;
$(\mu, \Sigma) \longleftarrow \text{kmeans}(Y, K = q+1)$;
$Q \longleftarrow q - 1$;
for $r = 1, \ldots, Q$ **do**
  for *all possible pairs* (i, j) *in* $\{1, \ldots, K\} \times \{1, \ldots, K\}$ **do**
    $P(i,j) \longleftarrow p\left(Y_i \cup Y_j | \mu_{ij}, \Sigma_{ij}\right)$;
  end for
  (i) either choose pair (i, j) with largest probability $P(i, j)$ and merge clusters i and j;
  (ii) or sample a pair (i, j) with probability $P(i, j)$ and merge clusters i and j;
  $q \longleftarrow q - 1$;
  $Y \leftarrow Y_{n\times q}$ (i.e. remove last dimension);
  update (μ, Σ);
  $K \leftarrow K - 1$;
end for

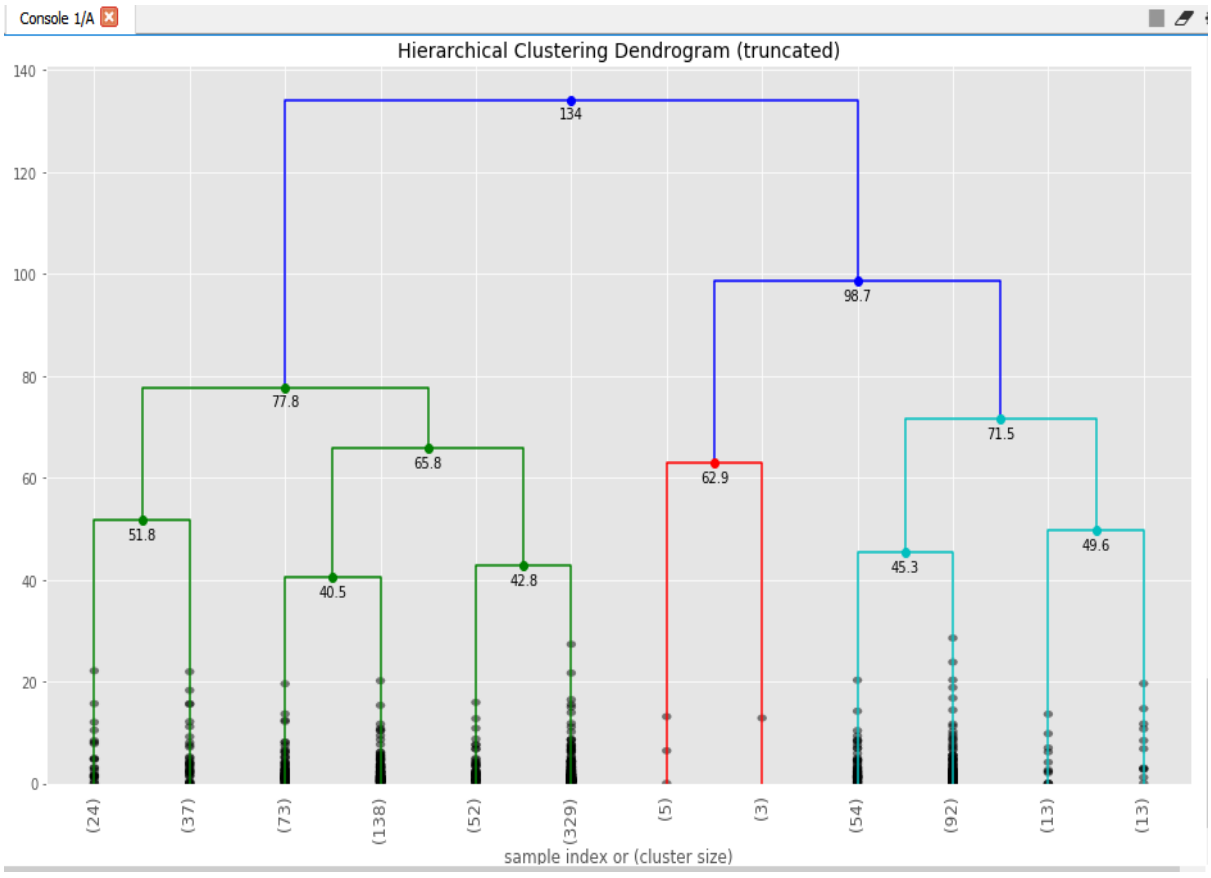
Sample Output Screen for Heirarchical Clustering (P2)

k_means.py | heirarchical.py | Console 1/A

```
import numpy as np
import pandas as pd
# Importing the dataset
data = pd.read_csv('xclara.csv')

data.head()
# Getting the values and plotting it
f1 = data['V1'].values
f2 = data['V2'].values
X = np.array(list(zip(f1, f2)))
import matplotlib.pyplot as plt

labels = range(1, len(X)-1)
plt.figure(figsize=(10, 7))
plt.subplots_adjust(bottom=0.1)
plt.scatter(X[:,0],X[:,1], label='True
for label, x, y in zip(labels, X[:, 0],
    plt.annotate(
        label,
        xy=(x, y), xytext=(-3, 3),
        textcoords='offset points', ha=
plt.show()
from scipy.cluster.hierarchy import den
from matplotlib import pyplot as plt

linked = linkage(X, 'complete')

labelList = range(1, len(X)+1)

plt.figure(figsize=(10, 7))
dendrogram(linked,
           orientation='top',
           labels=labelList,
           distance_sort='descending',
           show_leaf_counts=True)
plt.show()
```




140
120
100
80
60
40
20
0


Console 1/A
Hierarchical Clustering Dendrogram (truncated)
140
120
100
80
60
40
20
0
134
98.7
77.8
71.5
65.8
62.9
51.8
49.6
45.3
42.8
40.5
(24)
(37)
(73)
(138)
(52)
(329)
(5)
(3)
(54)
(92)
(13)
(13)
sample index or (cluster size)

Algorithm: Apriori (P3)

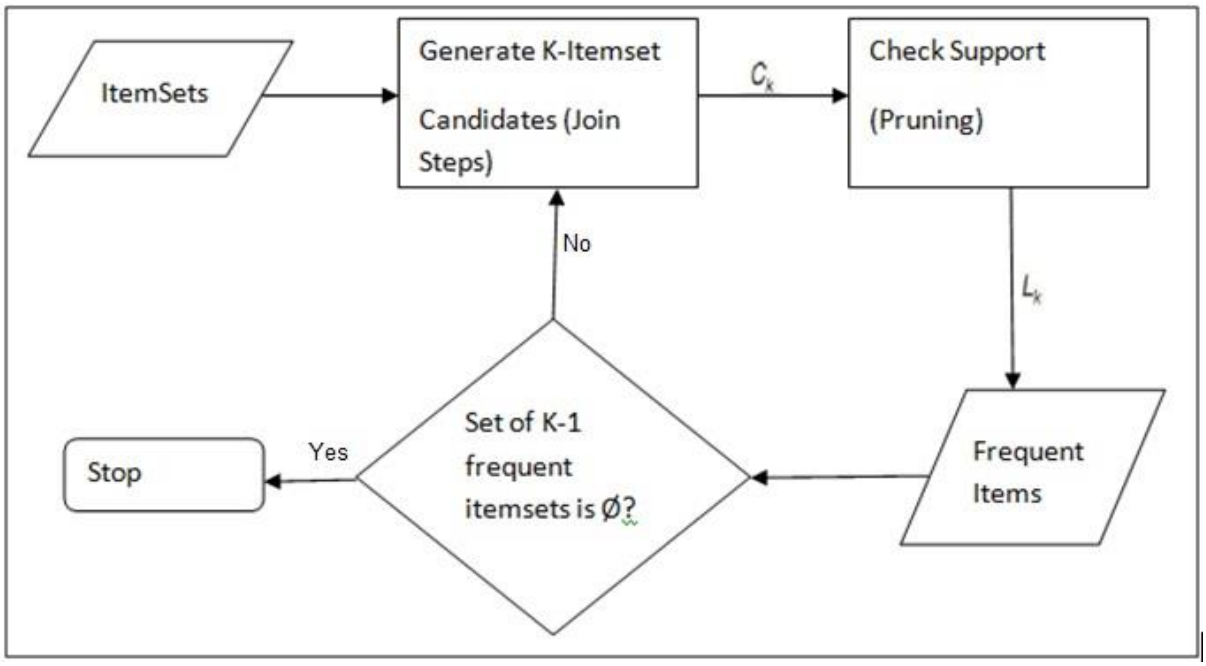
Apriori utilizes breadth-first search and a Hash tree structure to tally hopeful thing sets productively. It creates applicant thing sets of length k from thing sets of length k-1. At that point it prunes the hopefuls which have a rare sub design. As indicated by the descending conclusion lemma, the applicant set contains all successive k-length thing sets. From that point forward, it filters the exchange database to decide visit thing sets among the competitors.

Algorithm 3: Apriori algorithm

F_1 = {frequent items of size 1};
for (k = 1; F_k !=ϕ; k++) do begin
    C_{k+1} = apriori-gen (F_k); // New candidates generated from F_k
    for all transactions t in database do begin
        C'_t = subset (C_{k+1},t); // Candidates contained in t
        for all candidate $c \in C'_t$ do
            $c.count++$; // Increment the count of all candidates in C_{k+1} that are contained in t
        end
        F_{k+1} = {$C \in C_{k+1} | c.count \geq$ minimum suport}
        //Candidates in C_{k+1} with minimum support
    end
end
Answer $\bigcup_k F_k$;

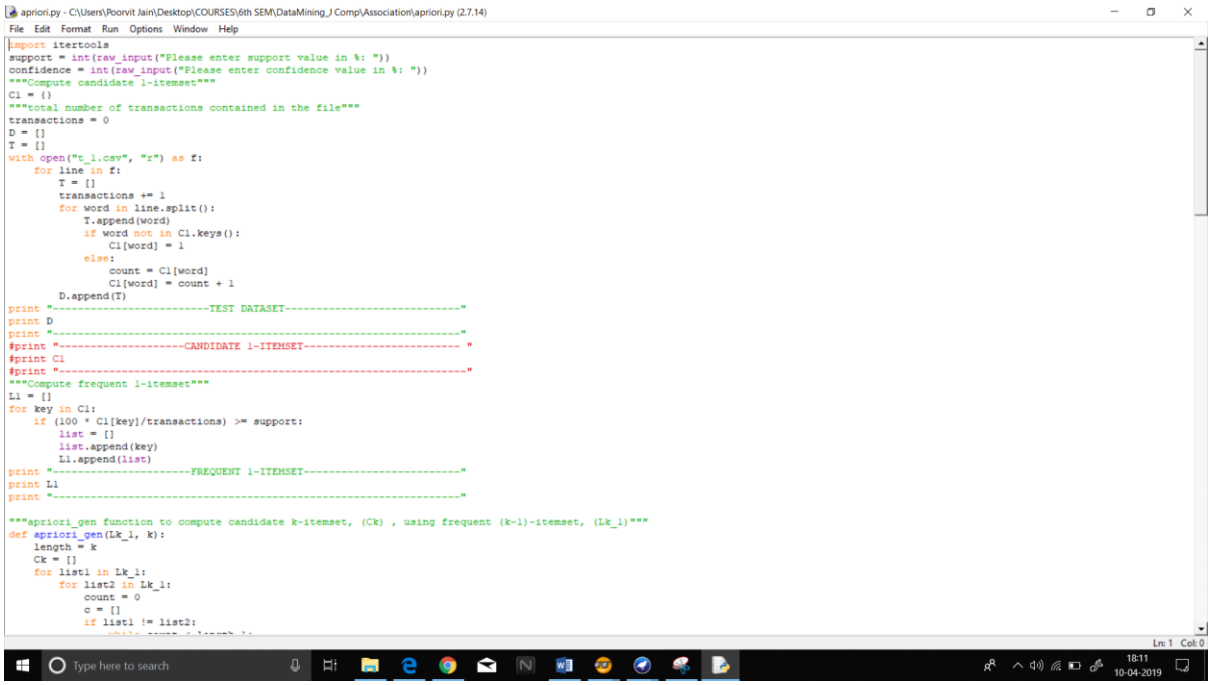
Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.

To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called Apriori property which helps by reducing the search space.

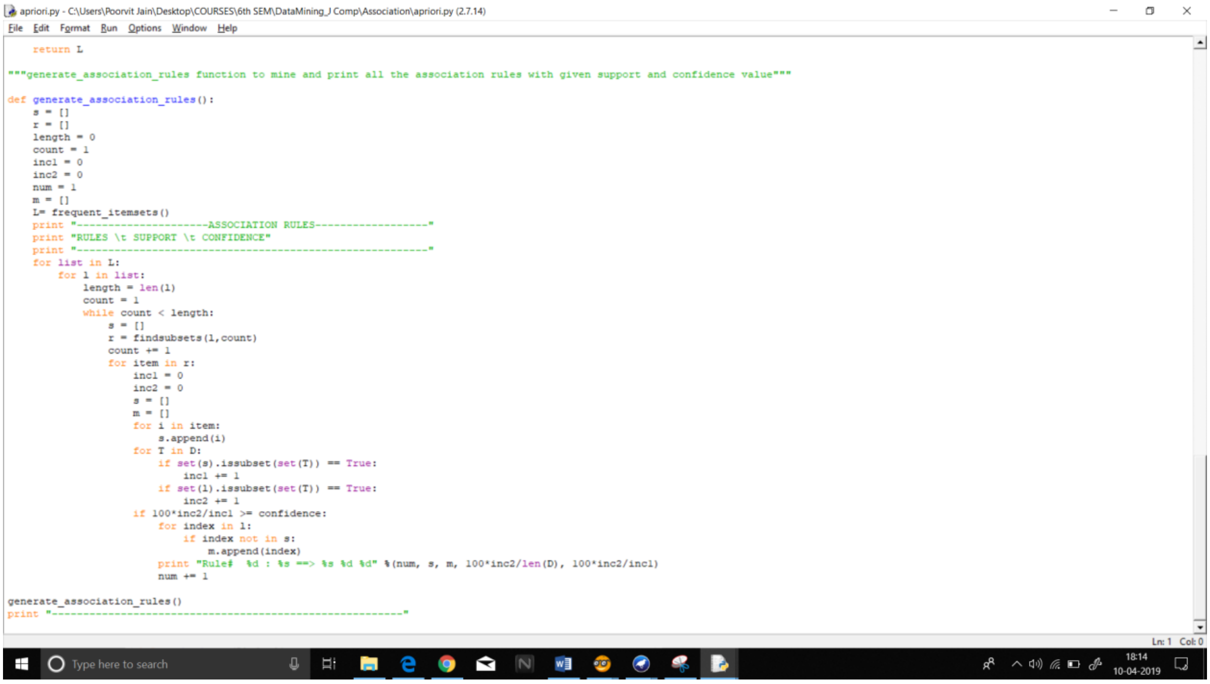


Sample Output Screen for Apriori (P3)

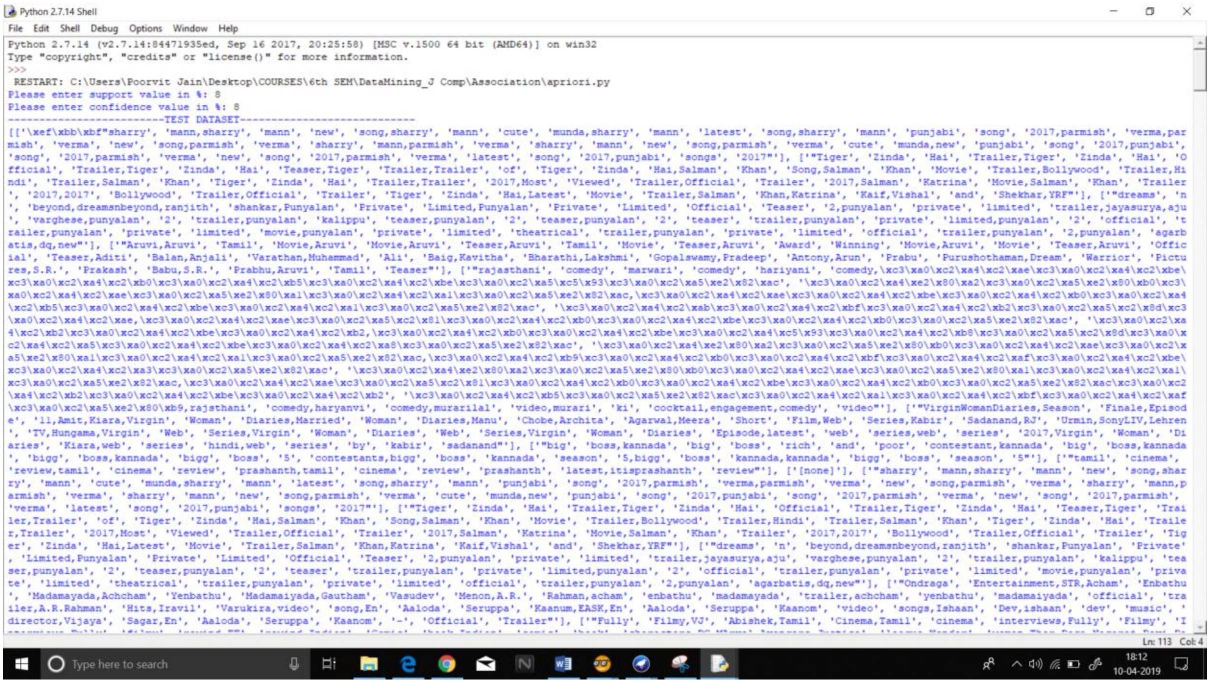
Code Used



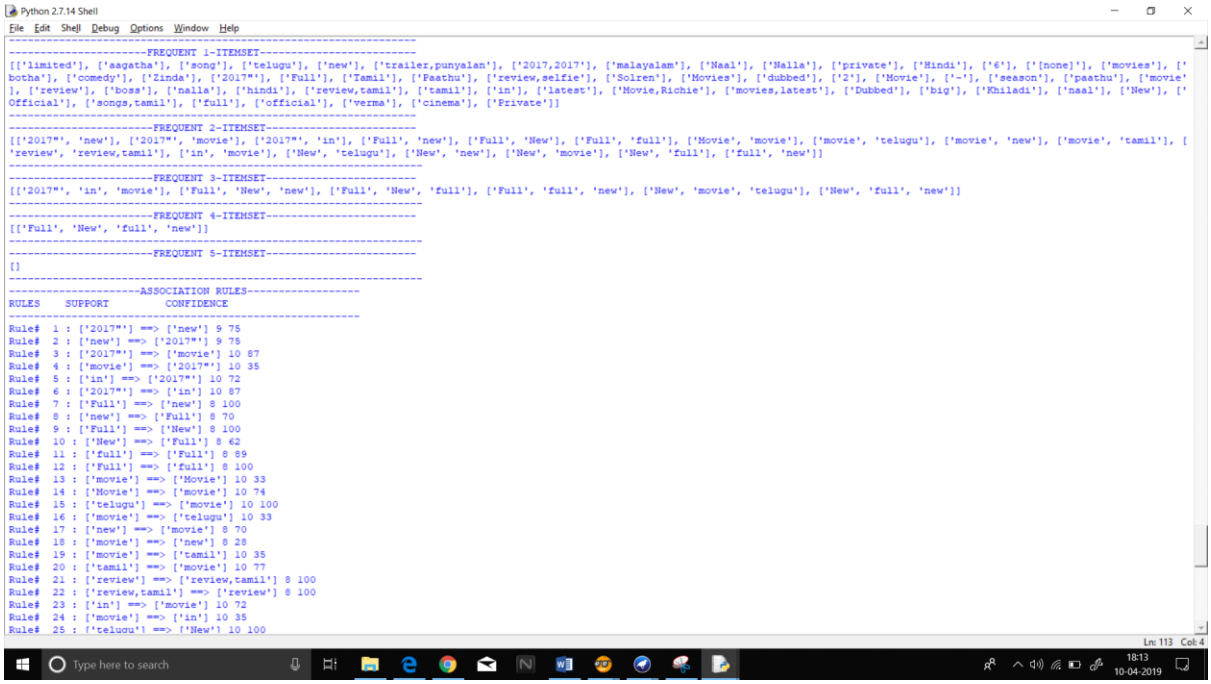
```
apriori.py - C:\Users\Poorvit Jain\Desktop\COURSES\6th SEM\DataMining_J Comp\Association\apriori.py (2.7.14)
File  Edit  Format  Run  Options  Window  Help
import itertools
support = int(raw_input("Please enter support value in %: "))
confidence = int(raw_input("Please enter confidence value in %: "))
"""Compute candidate 1-itemset"""
C1 = {}
"""total number of transactions contained in the file"""
transactions = 0
D = []
T = []
with open("t_1.csv", "r") as f:
    for line in f:
        T = []
        transactions += 1
        for word in line.split():
            T.append(word)
            if word not in C1.keys():
                C1[word] = 1
            else:
                count = C1[word]
                C1[word] = count + 1
        D.append(T)
print "-------------------------TEST DATASET--------------------------"
print D
print "---------------------------------------------------------------"
#print "--------------------CANDIDATE 1-ITEMSET------------------------ "
#print C1
#print "---------------------------------------------------------------"
"""Compute frequent 1-itemset"""
L1 = []
for key in C1:
    if (100 * C1[key]/transactions) >= support:
        list = []
        list.append(key)
        L1.append(list)
print "----------------------FREQUENT 1-ITEMSET-------------------------"
print L1
print "---------------------------------------------------------------"

"""apriori_gen function to compute candidate k-itemset, (Ck) , using frequent (k-1)-itemset, (Lk_1)"""
def apriori_gen(Lk_1, k):
    length = k
    Ck = []
    for list1 in Lk_1:
        for list2 in Lk_1:
            count = 0
            c = []
            if list1 != list2:
```



Dataset Used



Freqent Itemsets



Association Rules

Conclusion/ Inferences

The original data had the following properties:

Dimension:

1. Rows 37352
2. Columns 13

Attributes:

Name	Type	Description
video_id	Alpha-numeric	Unique id of 11 alpha-numeric characters
trending_date	Date	Trending date of a video
title	Text	Title of the video
channel_title	Text	Title of the channel
category_id	Number	Category of the video
publish_time	Time	Date and time of publishing video
tags	Text	Tags associated with each video
views	Number	Number of views acquired
likes	Number	Number of likes acquired on the video
dislikes	Number	Number of dislikes given for the video
comment_count	Number	Number of comments acquired
comments_disabled	Boolean	True/False
ratings_disabled	Boolean	True/False

By applying Decision Tree, Apriori, K-means and hierarchical clustering helped us in

1. Finding Category of a new video so as to maximize the number of views.

2. Clustering the dataset based on their category and country.

3. Dependency of one category on another.

We could also find the dependency of tags over each other by applying apriori over the tags collected from dataset (UCI Repository)

We were also able to test the accuracy and were able to visualize the results in the form of graph using:

1) Orange
2) Weka
3) Spyder